**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр КОВАЛЬ
«___»_____2020 р.

# Дипломна робота

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»**
**спеціальності 122 «Комп'ютерні науки та інформаційні технології»**
**на тему: «**Інструментальні засоби ідентифікації транспортних засобів**»**

Виконав:
студент IV курсу, групи ТМ-62
Кобець Ілля Олександрович _____
Керівник:
доцент, кандидат економічних наук
Гусєва Ірина Ігорівна _____
Рецензент:
доцент, кандидат технічних наук
Веремійчук Юрій Андрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

**Національний технічний університет України**
**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший рівень
Напрям підготовки 122 Комп’ютерні науки та інформаційні технології
Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

# ЗАВДАННЯ
**на дипломну роботу студенту**

_____Кобцю Іллі Олександровичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи Інструментальні засоби ідентифікації транспортних засобів керівник роботи Гусєва Ірина Ігорівна, доцент, кандидат економічних наук
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Python, TensorFlow, SQL, Java, Android
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Провести аналіз предметної області. Створити модель машинного навчання для класифікації транспортних засобів. Розробити андроїд додаток для визначення режиму транспортування в реальному часі. _____
5. Перелік ілюстративного матеріалу
“Тема”,”Постановка задачі”,”Задачі ідентифікації транспортних засобів”,”Методи ”,”Структура датасету”,”Загальна архітектура системи”,”Архітектура андроїд додатку”,”Цикл роботи”,”Опис БД”,”Підготовка даних”,”Опис роботи інтерфейсу додатку”,”Приклад роботи”,”Висновок”.
6. Дата видачі завдання ”_11_”___жовтня___2020 р.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітки |
|---|---|---|---|
| 1. | Затвердження теми роботи | 8.02.20 | |
| 2. | Вивчення та аналіз задачі | 13-19.04.20 | |
| 3. | Розробка архітектури та загальної структури системи | 20-26.04.20 | |
| 4. | Розробка структур окремих підсистем | 27.04-03.05.20 | |
| 5. | Програмна реалізація системи | 04-13.05.20 | |
| 6. | Оформлення пояснювальної записки | 14-16.05.20 | |
| 7. | Захист програмного продукту | 17.05.20 | |
| 8. | Передзахист | 12.06.20 | |
| 9. | Захист | 15.06.20 | |

Студент _____ _____Ілля КОБЕЦЬ_____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Ірина ГУСЄВА_____
(підпис) (прізвище та ініціали,)

# АНОТАЦІЯ

Метою цієї роботи є створення інструментального засобу ідентифікації транспортних засобів.

Створено модель класифікації транспортних засобів на основі даних з сенсорів та андроїд додаток для визначення режиму транспортування.

Загальний обсяг роботи: 51 сторінка, 28 ілюстрацій та 6 бібліографічних найменувань.

Ключові слова: ідентифікація транспортних засобів, андроїд додаток, машинне навчання, мобільні сенсори, розпізнавання активності.

# ABSTRACT

The purpose of this work is to create transportation mode detection tool.

A model of vehicle classification based on data from sensors and an android application for determining the mode of transportation has been created.

Total volume of work: 51 pages, 28 illustrations and 6 references.

Key words: transportation mode detection, tools, android app, machine learning, mobile sensors, activity recognition.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

**GPS** — Global Positioning System, система глобального позиціонування.

**БД** — база даних.

**API** — Application Program interface.

**GSM** — Global System for Mobile Communications.

**NN** — Neural network.

**AR** — Activity recognition.

**TMD** — Transportation mode detection.

**AR** — Machine Learning.

**IDE** — Integrated Development Environment.

# ВСТУП

Можливості мобільних телефонів зростають із розвитком апаратних та програмних технологій. Особливо датчики на смартфонах дозволяють збирати різноманітну інформацію. Таким чином, за допомогою смартфонів розпізнавання людської діяльності та виявлення режиму транспорту (TMD) стають основними напрямами досліджень за останнє десятиліття.

Ідентифікація транспортних засобів, або визначення режиму транспорту важлива для надання мобільним користувачам різноманітних послуг. Наприклад, орієнтовані на стан рекламні повідомлення, планування міського транспорту, моніторинг здоров'я.

Майже всі мобільні телефони в даний час містять датчики, які дозволяють фіксувати важливу контекстну інформацію. З цієї причини одним із ключових датчиків, використовуваних контекстними програмами, є мобільний телефон, який став центральною частиною життя користувачів. Визначення режиму транспортування використовує датчики, включаючи GPS, акселерометр, гіроскоп і магнітометр на носимих предметах, які періодично перевозяться під час транспортування. Можна використовувати деякі зовнішні датчики, але найпростіший спосіб отримання даних - це використання смартфонів або смарт-годинників. Таким чином, багато дослідників збирають необхідні дані за допомогою цих пристроїв, які здатні обробляти сенсорні дані для прийняття рішення.

Одне з перших досліджень в сфері інструментальних засобів ідентифікації транспортних засобів було проведено Ієном Андерсоном та Хенком Мюллером у 2006 році Вони намагались розпізнати чи людина стоїть, ходить чи їздить на автомобілі, використовуючи швидкість передачі сигналів GSM для мобільних телефонів. Відповідно до їх підходу, швидкість зміни сигналу або зміна базової станції допомагає визначити активність. У своєму дослідженні вони намагалися довести, що сигнал не змінюється часто, якщо користувач стоїть, або він змінюється дуже швидко,

якщо людина їздить на машині. У цій роботі були втілені k-NN та модель прихованого Маркова, а рівень успішності становив лише 80%, при наявності всього 3 класів.

Розпізнавання режиму транспортування користувача може розглядатися як завдання AR. Розпізнавання режиму транспорту може надати інформацію про контекст для покращення програм та покращення користувальницької роботи. Ці данні будуть корисними для багатьох різних програм, таких як профілювання пристроїв, моніторинг стану дорожнього руху, охорона здоров'я, підтримка подорожей тощо.

Зі збільшенням попиту на різні додатки ефективні методи класифікації транспорту стали необхідними, хоча і складними. Це питання було широко вивчено на основі зворотного зв’язку з GPS. Однак GPS вимагає прямого огляду між пристроями та супутниками. Таким чином, він не підходить для внутрішніх або міських середовищ. Крім того, методи на основі GPS споживають значну потужність і можуть не працювати ефективно для таких заходів, як біг та ходьба. Багато робіт вивчають класифікацію видів транспорту, більшість з них покладаються на дані GPS або інформацію про бездротову мережу. В останні роки було розроблено кілька методів на основі датчиків, вбудованих у смартфони, такі як акселерометр, магнітометр, гіроскоп, атмосферний тиск та ін. Використання цих датчиків було визнано хорошим підходом для визначення положення тіла та режимів руху. Однак дослідження сенсорних підходів щодо класифікації видів транспорту обмежені.

Найбільш ефективним інструментом в вирішенні проблеми ідентифікації транспортних засобів є великий підрозділ штучного інтелекту – машинне навчання. В даній роботі будуть використані та створення моделі машинного навчання та порівняні такі алгоритми класифікації Random forest (Випадковий ліс), Дерево ухвалення рішень(Decision tree), Нейронна мережа(Neural Network).

В останні роки концепція глибокого навчання привернула значну увагу. Численні дослідження підтвердили, що модель глибокого навчання, сформована шляхом складання декількох шарів неглибоких структур, має кращу здатність представити особливості і, відповідно, може ефективніше вирішувати завдання

нелінійної та високої складності. Моделювання великих даних за допомогою моделі глибокого навчання було визнано ефективним в багатьох сферах, таких як прогнозування потоку руху, класифікація типів транспортних засобів.

В дослідженнях укладаючи кілька шарів нейронів з оптимізованими вагами, вченими була запропонована глибока нейронна мережа (DNN) яка може ефективно моделювати нелінійну функцію між послідовними датчиками зондування та міченими атрибутами. Таким чином, може точно визначається режим транспортування на основі даних мобільних сенсорів.

Потужність сучасних смартфонів зростає в геометричній прогресії що дозволяє швидко аналізувати данні і відображати результат. Попри це ефективність та ємкість акумуляторів смартфонів не розвивається відповідно швидко. Що призводить до проблеми знаходження балансу між енергоефективністю та точністю визначення режиму транспортування. А саме в частоті збирання даних з сенсорів. Також ця проблема актуальна при зборі даних в набори для аналізу. Окрім того частота впливає на розмір зібраних даних.

# 1. Задача ідентифікації транспортних засобів.

Мета: створення програмного забезпечення для класифікації режимів пересування людини використовуючи методи машинного навчання та дані сенсорів андроїд смартфону.

Задачі:

- Зібрати дані з мобільних сенсорів для використання.
- Створити модель машинного навчання для класифікації режиму пересування користувача.
- Розробити базу даних для зберігання даних сенсорів телефону, та інших необхідних даних для роботи андроїд додатку.
- Створити андроїд додаток для розпізнавання режиму пересування користувача в реальному часі та відображенні історію використаних режимів пересування на основі даних мобільних сенсорів.

Інструментальний засіб може бути використаний розробниками для застосування в багатьох сферах та отримання рішення багатьох актуальних проблем. Наприклад, які загальні дороги вони використовують під час подорожей та на яких саме транспортних засобах? Відповіді на такі запитання можуть допомогти уряду зрозуміти потреби міста та допоможуть підвищити рівень щастя громадян. Виявлення типу транспорту може також дозволяти нам показувати різну рекламу для оцінених користувачів. Наприклад, якщо людина їде в машині, можуть бути показані сервіси з обслуговування автомобілів  або якщо людина подорожує в автобусі, книги або планшетні ПК. Також інші сфери, такі як системи розміщення та позиціонування , інтелектуальні сервіси, засновані на локації, є потенційними користувачами інструментальних засобів виявлення режиму транспортування.

Вхідною інформацією є данні зібрані з сенсорів мобільного телефону.

Вихідною інформацією є передбачення режиму пересування користувача.

# 2. АНАЛІЗ ПРОБЛЕМИ СТВОРЕННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ІДЕНТИФІКАЦІЙ ТРАНСПОРТНИХ ЗАСОБІВ

## 2.1 Загальний опис задачі ідентифікації транспортних засобів

Задача ідентифікації транспортних засобів відноситься до Human Activity Recognition(HAR). Полягає в класифікації транспортних засобів користувача тобто режиму пересування в певний момент часу наприклад пішки, на машині тощо.

Для вирішення задачі зазвичай використовують GPS, але сучасні дослідження підтверджують ефективність використання групи мобільних сенсорів.

Було розроблено багато засобів на основі GPS. Однак GPS вимагає прямого зв’язку між пристроями та супутниками, не підходить для внутрішніх або міських середовищ. Крім того, методи на основі GPS велику кількість зарядку батареї і можуть не працювати ефективно для такої активності, як біг чи ходьба. Найефективнішими для вирішення цієї задачі є методи машинного навчання з використанням групи мобільних сенсорів.

## 2.2 Опис мобільних сенсорів

Для визначення способу транспортування з використанням смартфонів, потрібно використовувати дані з різних вбудованих сенсорів мобільного телефону. Більшість сучасних пристроїв мають датчики, котрі вимірюють умови навколишнього середовища, орієнтацію та інше. Сучасні мобільні телефони надають ці дані з високою точністю.

Датчики руху включають в себе акселерометри, датчики тяжіння, гіроскопи, тощо. В свою чергу датчики положення включають датчики орієнтації та магнітометри. Екологічні датчики це барометри, фотометри та термометри.

Акселерометри здатні вимірювати фізичний рух мобільного телефону чи іншого предмету. Тобто вони вимірюють силу прискорення до пристрою на всіх трьох фізичних осях, включаючи силу тяжіння. Акселерометри в основному використовуються для орієнтації в смартфонах. Основна особливість, яка робить цей датчик привабливим, - це низьке споживання енергії.

Гіроскоп вимірює швидкість обертання пристрою навколо кожної з трьох фізичних осей. Гіроскопи характеризуються низьким споживанням енергії, але схильні до накопичення помилок внаслідок значних помилок калібрування, електронного шуму та температури.

Магнітометр вимірює навколишнє геомагнітне поле для всіх трьох фізичних осей. Це забезпечує орієнтацією щодо магнітного поля Землі.

Датчик глобальної системи позиціонування (GPS) забезпечує інформацію про положення та швидкість користувача, який вимірюється залежно від відстані мобільного телефону до кожного з ряду супутників. Недоліками є залежність від фізичної відстані до супутників та високе енергоспоживання.

Сигнали GSM використовуються телефоном для дзвінків та передачі даних. Найпоширеніший. Завдяки роботі принципу роботи GSM мобільні телефони можна відстежувати у зовнішніх та внутрішніх умовах. Точність різна залежно від розміру GSM зон(від 50 до 200 метрів), але може погіршуватися ще більше в районах з поганим сигналом.

## 2.3 Опис режимів пересування

Режими транспортування(пересування) можливо класифікувати грубо на 2 класи моторизовані та на не моторизовані. Моторизовані режими пересування представляють автомобілі, мотоцикли, вантажівки, автобуси, трамваї, метро та потяги. Немоторизовані режими пересування включають ходьбу, біг та їзду на велосипеді.

## 2.4 Класифікація методів ідентифікації транспортних засобів.

Генеративні алгоритми навчаються без вчителя тобто умовно можна розділити на дві частини, частину яка відповідає на питання та на частину яка перевіряє відповідь. Популярні генеративні алгоритми цієї групи відносяться Naive Bayes, Bayesian Networks, Mexure Models.

Дискримінаційні алгоритми на відмінну від генеративних моделюються тільки з використання наявних даних. Популярні дискримінаційні алгоритми: Decision Tree, Support Vector Machines, Random Forest, Nearest Neighbor, Neural Network, Clustering.

На (рисунку 2.1) зображено дерево класифікації методів.

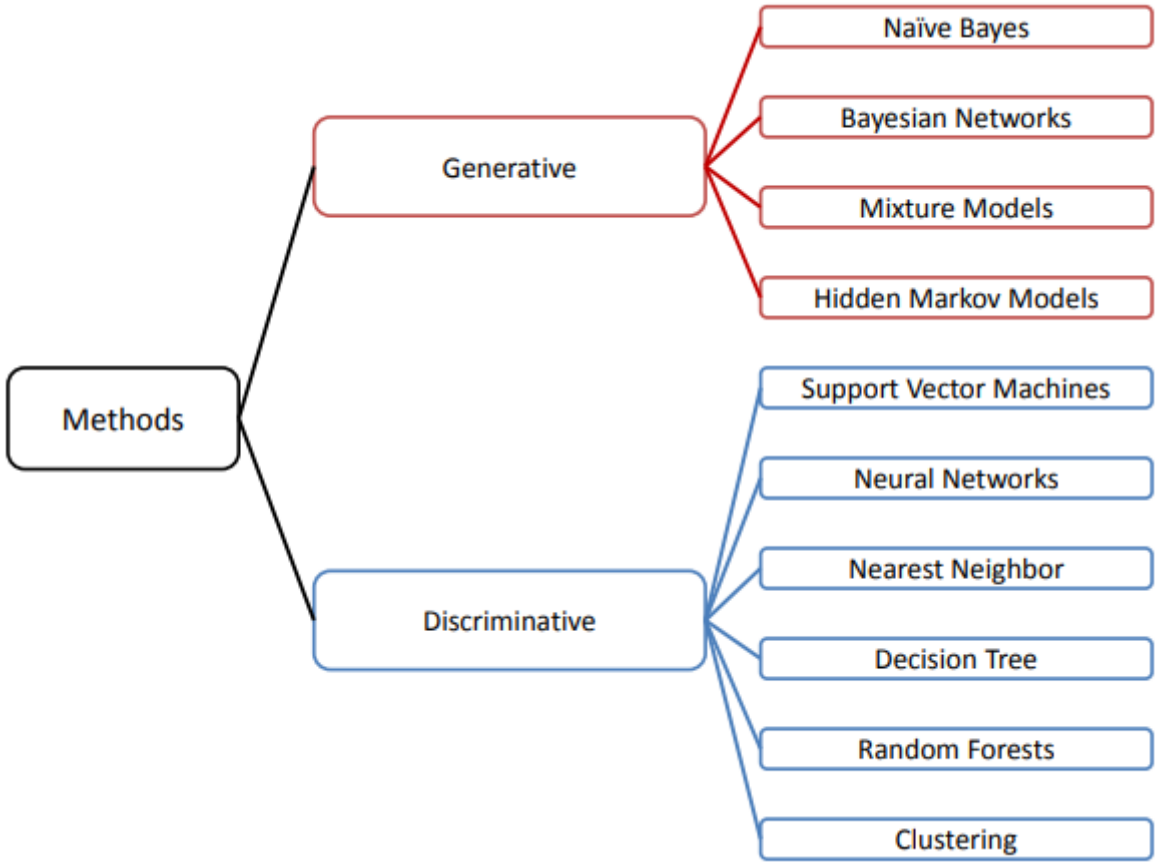


Рисунок 2.1 – дерево класифікації методів.

## 2.5 Опис роботи методів машинного навчання

### Decision Tree

У машинному навчанні дерева рішень - це техніка створення моделей прогнозування. Вони називаються деревами рішень, тому що передбачення слідує за декількома гілками рішення "якщо ... тоді ..." - подібно до гілок дерева. Якщо ми уявляємо, що ми починаємо з вибірки, для якої ми хочемо передбачити клас, ми б почали внизу дерева і подорожували вгору по стовбуру, поки не прийдемо до першої розгалуженої гілки. Цей розкол можна розглядати як особливість машинного навчання, скажімо, це був би "вік"; Тепер ми приймаємо рішення про те, яку галузь слід використовувати: "якщо наш зразок має вік більше 30 років, продовжуйте по лівій гілці, а далі продовжуйте по правій гілці". Це ми робимо, поки не прийдемо до наступної гілки та повторимо той самий процес прийняття рішень, поки не буде більше гілок перед нами. Ця кінцева точка називається аркушем, і в деревах рішень буде представлений кінцевий результат: передбачуваний клас або значення.

На кожній гілці знаходять пороги функцій, які найкраще розділяють зразки локально. Найпоширенішими показниками для визначення «найкращого розколу» є домішка джині та отримання інформації для завдань класифікації та зменшення дисперсії для регресії.

Дерева рішень дуже легко уявити та зрозуміти, оскільки вони дотримуються методу прийняття рішень, який дуже схожий на те, як ми, люди, приймаємо рішення: з ланцюжком простих правил. Однак вони не дуже надійні, тому що не добре узагальнюють невидимі зразки.

### Random forest

Дерева рішень дуже чутливі до даних, на яких вони навчаються - невеликі зміни в навчальному наборі можуть призвести до істотно різних структур дерев. Випадковий ліс використовує це, дозволяючи кожному окремому дереву випадковим чином вибирати з набору даних із заміною, в результаті чого виникають різні дерева. Цей процес відомий як пакування.

Зауважте, дані тренувань не розбиваються на менші шматки. Якщо у нас є зразок розміру N, ми все одно годуємо кожне дерево навчальним набором розміром N (якщо не вказано інше). Але замість вихідних даних про навчання ми беремо випадкову вибірку розміру N із заміною. Наприклад, якщо б наші дані про навчання були [1, 2, 3, 4, 5, 6], то ми могли б дати одному з наших дерев такий перелік [1, 2, 2, 3, 6, 6]. Зауважте, що обидва списки мають шість завдовжки і що "2" та "6" повторюються у випадково вибраних навчальних даних, які ми надаємо нашому дереву (тому що ми вибірки із заміною).

Розбиття вузлів у моделі ґрунтується на випадковому наборі функцій для кожного дерева.

Випадковість функції - у звичайному дереві рішень, коли настав час розділити вузол, ми розглядаємо всі можливі функції та вибираємо ту, яка виробляє найбільше розмежування між спостереженнями у лівому вузлі порівняно з тими, що знаходяться у правому вузлі. Навпаки, кожне дерево у випадковому лісі може вибирати лише випадкову підмножину ознак. Це зумовлює ще більшу варіативність дерев у моделі та, в кінцевому рахунку, призводить до меншої кореляції між деревами та більшої диверсифікації.

### Neural Network

Нейронні мережі - це набір алгоритмів, розроблених за принципом роботи мозку людини, для розпізнавання шаблонів. Вони інтерпретують сенсорні дані через

своєрідне машинне сприйняття, маркування або групування вихідних даних. Шаблони, які вони розпізнають, є чисельними та містяться у векторах.

Нейронні мережі використовують для розв’язання задач кластеризації та класифікації. Ви можете розглядати їх як шар кластеризації та класифікації поверх даних, які ви зберігаєте та керуєте. Вони допомагають групувати непомічені дані за подібністю серед прикладних входів, і вони класифікують дані, коли у них є мічений набір даних для навчання. (Нейронні мережі також можуть витягувати функції, що подаються на інші алгоритми кластеризації та класифікації; тому ви можете думати про глибокі нейронні мережі як компоненти більш великих додатків машинного навчання, що включають алгоритми для навчання, класифікації та регресії.)

## 2.6 Огляд датчиків мобільних андроїд пристроїв

Більшість пристроїв на базі Android мають вбудовані датчики, які вимірюють рух, орієнтацію та різні екологічні умови. Ці датчики здатні забезпечувати необроблені дані з високою точністю та точністю, і вони корисні, якщо ви хочете контролювати переміщення або позиціонування тривимірних пристроїв або хочете відстежувати зміни в навколишньому середовищі біля пристрою. Наприклад, гра може відслідковувати показання від датчика сили тяжіння пристрою для виведення складних жестів і рухів користувача, таких як нахил, тремтіння, обертання або розгойдування. Аналогічно, програма для погоди може використовувати датчик температури і датчик вологості пристрою для обчислення та повідомлення температури точки роси, або дорожня програма може використовувати датчик геомагнітного поля та акселерометр для повідомлення про підшипник компаса.

Платформа Android підтримує три широкі категорії датчиків:

Датчики руху: ці датчики вимірюють сили прискорення та сили обертання вздовж трьох осей. До цієї категорії належать акселерометри, датчики гравітації, гіроскопи та датчики обертального вектора.

Екологічні датчики: ці датчики вимірюють різні параметри навколишнього середовища, такі як температура та тиск навколишнього повітря, освітленість та вологість. До цієї категорії належать барометри, фотометри та термометри.

Датчики положення: ці датчики вимірюють фізичне положення пристрою. До цієї категорії належать датчики орієнтації та магнітометри.

Ви можете отримати доступ до датчиків, наявних на пристрої, та отримати необроблені дані сенсорів за допомогою Android sensor framework. Фреймворк містить декілька класів та інтерфейсів, які допомагають виконувати найрізноманітніші завдання, пов'язані з датчиком.

Наприклад, ви можна використовувати фреймворк датчика, щоб зробити наступне:

- визначити доступні датчики на пристрої
- визначення індивідуальних можливостей датчика
- отримати необроблені дані датчика.

**Android sensor framework**

Android sensor framework дозволяє отримати доступ до багатьох типів датчиків. Деякі з цих датчиків є апаратними, а деякі - програмними. Апаратні датчики - це фізичні компоненти, вбудовані в слухавку або планшетний пристрій. Вони отримують свої дані шляхом прямого вимірювання конкретних властивостей навколишнього середовища, таких як прискорення, напруженість геомагнітного поля або зміна кута. Датчики на основі програмного забезпечення не є фізичними пристроями, хоча вони імітують апаратні сенсори. Датчики на основі програмного забезпечення отримують свої дані від одного або декількох апаратних сенсорів, а іноді їх називають віртуальними датчиками або синтетичними датчиками. Датчик лінійного прискорення і датчик сили тяжіння - приклади датчиків на основі програмного забезпечення.

Небагато пристроїв на базі Android мають кожен тип датчиків. Наприклад, у більшості телефонів і планшетів є акселерометр і магнітометр, проте в меншій кількості пристроїв є барометри або термометри. Також пристрій може мати більше одного датчика заданого типу. Наприклад, пристрій може мати два датчика сили тяжіння, кожен з яких має різний діапазон.

Датчики, які підтримуються платформою Android:

- TYPE_ACCELEROMETER
- TYPE_AMBIENT_TEMPERATURE
- TYPE_GRAVITY
- TYPE_GYROSCOPE
- TYPE_LIGHT
- TYPE_LINEAR_ACCELERATION
- TYPE_MAGNETIC_FIELD
- TYPE_ORIENTATION.
- TYPE_PRESSURE
- TYPE_PROXIMITY
- TYPE_RELATIVE_HUMIDITY
- TYPE_ROTATION_VECTOR TYPE_TEMPERATURE

**Інтерфейси та пакети Android sensor framework**

SensorManager: цей клас використовується для створення примірника служби датчика. Цей клас надає різні методи доступу до датчиків та їх переліку, реєстрації та відреєстрації слухачів сенсорних подій та отримання інформації про орієнтацію. Цей клас також містить кілька констант датчиків, які використовуються для повідомлення точності датчика, встановлення швидкості збору даних та калібрування датчиків.

Sensor: цей клас використовується для створення примірника конкретного датчика. У цьому класі представлені різні методи, які дозволяють визначити можливості датчика.

SensorEvent: система використовує цей клас для створення об'єкта події датчика, який надає інформацію про датчик події. Об'єкт події датчика включає таку інформацію: необроблені дані датчика, тип датчика, який генерував подію, точність даних та часову позначку події.

SensorEventListener: інтерфейс використовується для створення двох методів зворотного виклику, які отримують сповіщення (сенсорні події), коли змінюються значення датчика або коли змінюється точність датчика.

**Доступність датчиків**

Хоча доступність датчика залежить від пристрою, він також може відрізнятися між версіями Android. Це тому, що сенсори Android були представлені протягом декількох релізів платформи. Наприклад, багато сенсорів було введено в Android 1.5 (API рівень 3), але деякі не були реалізовані і не були доступні для використання до Android 2.3 (API рівень 9). Так само було введено кілька датчиків в Android 2.3 (API рівень 9) та Android 4.0 (API рівень 14). Два датчики застаріли і замінені новими, кращими датчиками.

# 3. ЗАСОБИ РОЗРОБКИ

## 3.1 Мова програмування Python

Python - інтерпретована мова програмування, належить до мов програмування високого рівня, має загального призначення. Мова програмування була створена Створена Гідо ван Россумом. В 1990 році була вперше випущена, Python має чітку філософію дизайну яка підкреслює читабельність коду використовуючи помітне використання структуризації пробілом. Об'єктно-орієнтований підхід мови та її мовні конструкції створені з метою допомогти програмістам писати логічний та чіткий код для проектів будь якого розміру. Мова програмування Python завдяки цьому вважається найкращою мовою програмування для навчання.

В мові Python реалізоване динамічне збирання сміття. Також мова підтримує багато парадигм програмування, включає такі парадигми: структуроване (зокрема, процедурне), об'єктно-орієнтоване і функціональне програмування.

Пітон почали створювати в 1985 роках як мову назаміну мови ABC. Python 2 був випущений у 1999 році, він представляє такі функції: розумні списка, збір сміття, здатність збирати еталонні цикли. Python3 який було випущено у 2008 році, значно розширив можливості мови, тому код Python3 не сумісний з Python2, що призвело до потреби адаптації великої кількості бібліотек до нової версії.

Підтримка мови програмування Python 2 була офіційно припинена у 2020 році (хоча спочатку планувалася на 2015 рік), а Python 2.7.18 - це останній випуск Python 2.7, а отже, останній випуск Python 2. Більше не буде випущено жодних патчів безпеки чи інших удосконалень. З закінченням терміну служби Python 2 підтримується лише Python 3.5.x та пізніші версії.

На даний момент 75 відскотків всього написаного коду на пайтон використовує версію Python 3.x, а 25 відсотків використовує версію Python 2.x

Інтерпретатори Python доступні для багатьох операційних систем. Глобальне співтовариство програмістів розробляє та підтримує CPython, відкритий вихідний

код. Некомерційна організація, програмний фонд Python, керує та спрямовує ресурси для розробки Python та CPython.

Python використовує так названий duck typing. Мова програмування дозволяє програмістам визначати власні типи за допомогую класів.

На (рисунку 3.1) зображена стандартна ієрархія типів Python.

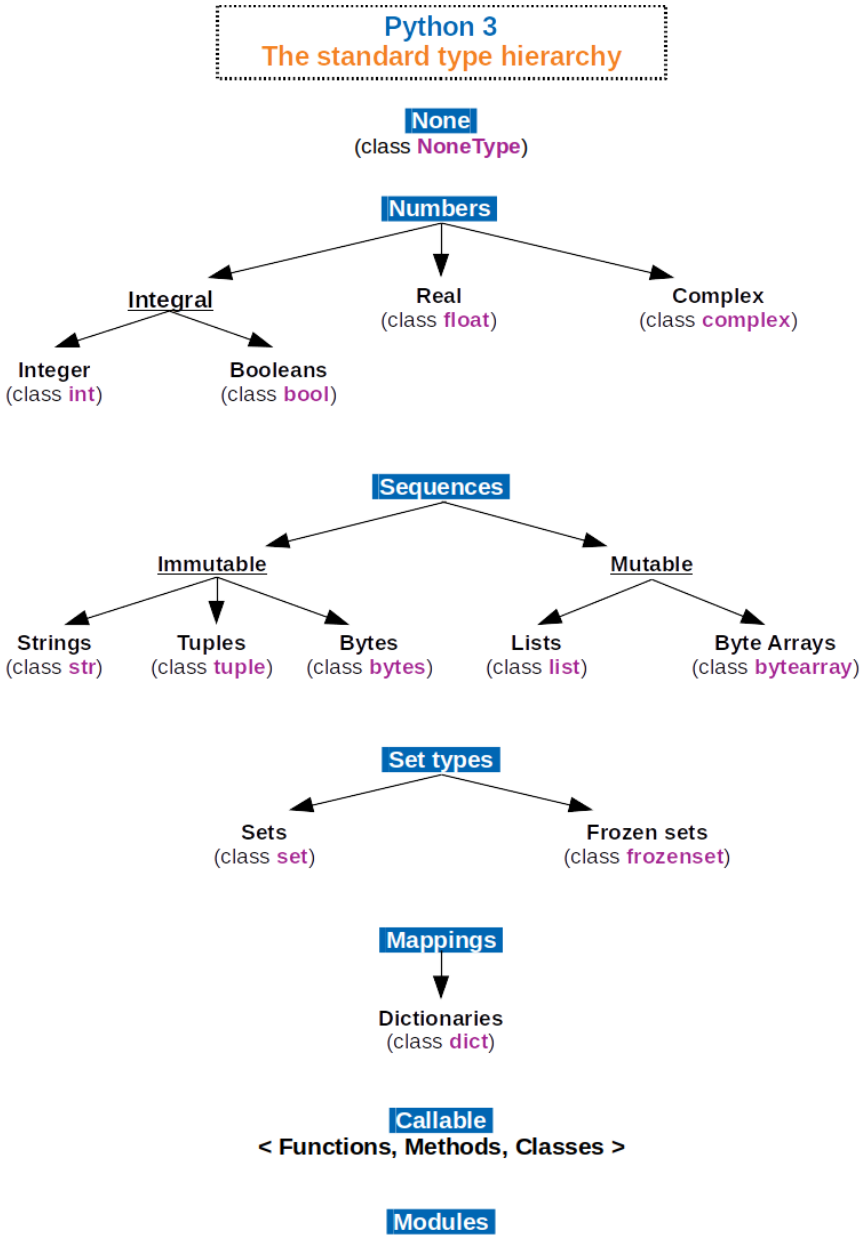


Рисунок 3.1 – стандартна ієрархія типів Python.

## 3.2 Бібліотека TensorFlow

TensorFlow - бібліотека Python для швидких чисельних обчислень, створених та випущених Google. Це фундаментальна бібліотека, яка може бути використана для створення моделей глибокого навчання безпосередньо або за допомогою бібліотек обгортки, які спрощують процес, побудований на TensorFlow.

Переваги TensorFlow:

- Легкий для вивчення;
- Легкий для використання;
- Потужний;
- API більше знайомі розробникам Python;
- Інтегрований Keras;

Архітектура TensorFlow зображена на (рисунку 3.2).

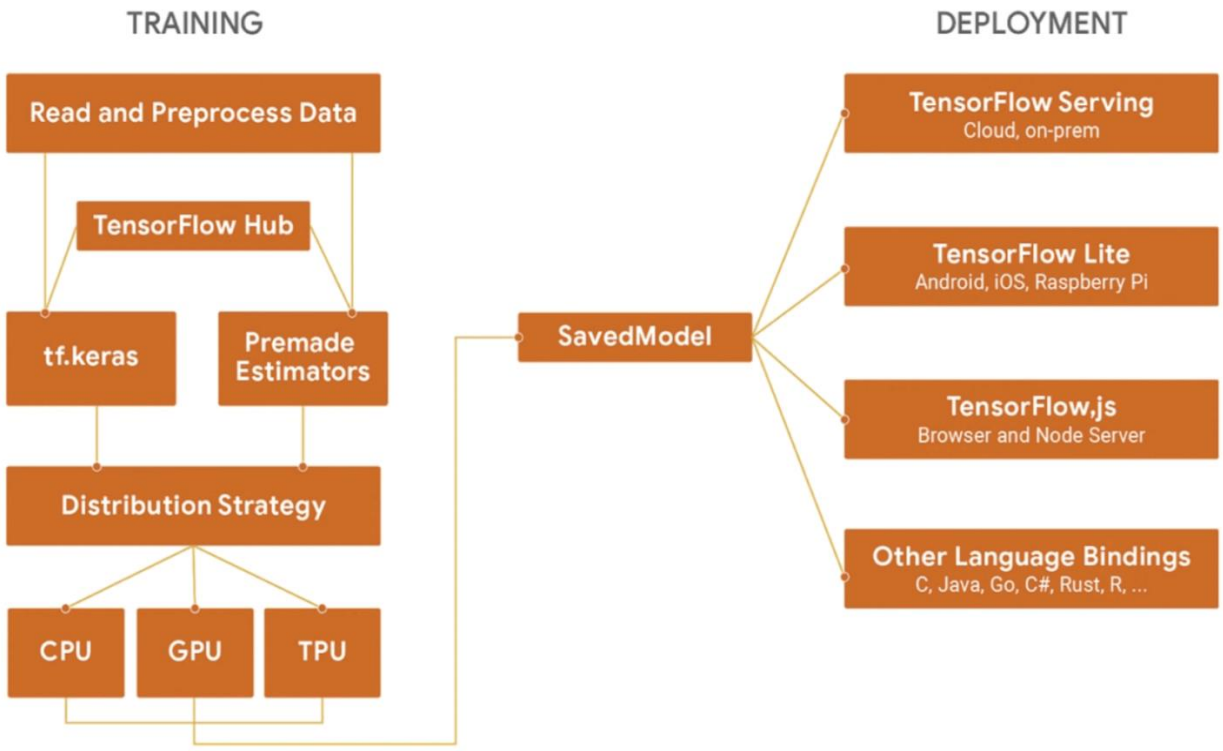


Рисунок 3.2 – Архітектура TensorFlow.

## 3.3 Бібліотека Scikit-learn

Scikit-learn (раніше scikits.learn і також відомий як sklearn) - це безкоштовна бібліотека машинного навчання для мови програмування Python. Він містить різні алгоритми класифікації, регресії та кластеризації, включаючи підтримуючі векторні машини, випадкові ліси, збільшення градієнта, k-засоби та DBSCAN, і призначений для взаємодії з числовими та науковими бібліотеками Python NumPy та SciPy.

## 3.4 Мова програмування Java

Java - це мова програмування яка відноситься до мов групи загального призначення, мова заснована на класах, має об'єктно-орієнтовану парадигму і створена для того щоб мати мінімум залежностей від реалізації. Це призначено для того, щоб розробники які користуються мовою додатків могли писати один раз, та написану програму запускати на будь якій платформі (WORA), отже зкомпільований код Java може бути ітерпритованим на всіх платформах, які підтримують Java без необхідності русурсо затратної перекомпіляції. Програми Java завжди компілюються в спеціальний в байт-код, який може працювати на всіх та будь яких віртуальних машинах Java (JVM) незалежно від основної оперативної системи використаної на комп'ютері розробника. Синтаксис Java сильно схожий на C і C ++, та попри це в ньому набагато менше низького рівня, ніж в представлених мовах. За станом на 2020 рік, Java є однією з найпопулярніших мов для використання, та застосовується відповідно до даних GitHub, особливо часто для веб-додатків та клієнт-серверів.

Джава було розроблена Джеймсом Гослінгом у компані Sun Mircosytems та випущена в 1996. Оригінальні та довідкові реалізатори Java-компілятори, віртуальні машини та бібліотеки класів були спочатку випущені Sun під власні ліцензії. Станом на травень 2007 року, відповідно до специфікацій процесу спільноти Java, Sun передала більшість своїх технологій Java відповідно до Загальної публічної ліцензії GNU. Тим часом, інші розробили альтернативні реалізації цих технологій Sun, такі як

GNU Compiler for Java (компілятор байт-кодів), GNU Classpath (стандартні бібліотеки) та IcedTea-Web (плагін браузера для аплетів).

## 3.5 Фреймворк TensorFlow Lite

TensorFlow Lite - це фреймворк з глибоким навчанням та відкритим кодом для імплементування готових моделай  машинного навчання.

Нова версія бібліотеки спеціально розроблена для навчання на невеликих пристроях. Серед переваг Lite-версії варто виділити:

- Легкість: забезпечує швидку ініціалізацію для навчання на невеликих пристроях;
- Багатоплатформність: навчання моделей можливо на великий кількості мобільних платформ, враховуючи Android та iOS;
- Швидкість: бібліотека оптимізована для використання на мобільних, та підтримує апаратне пришвидшення.

Компонент TF Lite:

- TensorFlow Model: навчена та збережена модель TensorFlow;
- TensorFlow Lite Converter: програма, конвертуюча модель у форматі TensorFlow Lite;
- TensorFlow Lite Model File: формат файлу моделі на основі FlatBuffers, оптимізований для максимальної швидкості та мінімального розміру.

На (рисунку 3.3) зображений короткий опис роботи фреймворку.

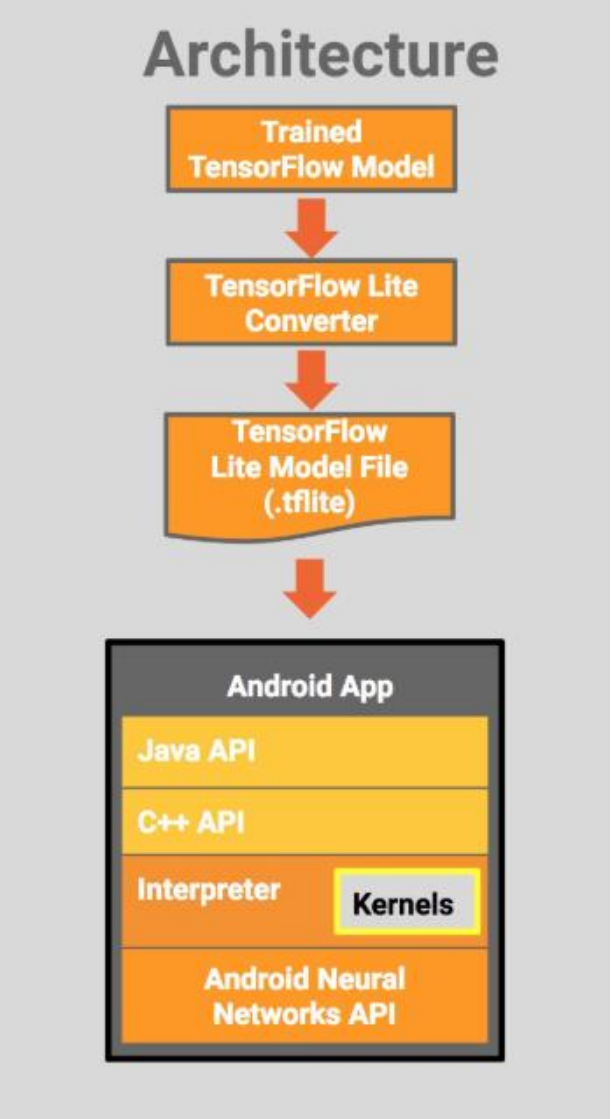


Рисунок 3.3 – схема роботи фреймворку.

## 3.6 Середовище розробки Android Studio

Android Studio - інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсована 16 травня 2013 року на конференції Google I/O.

Дана IDE перебувала у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні

2014 року, тоді ж припинилася підтримка плагіна Android Development Tools (ADT) для Eclipse.

Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, - офіційне засіб розробки Android додатків. Дане середовище розробки доступна для Windows, macOS і Linux. 17 травня 2017, на щорічній конференції Google I / O, Google анонсував підтримку мови Kotlin, використовуваного в Android Studio, як офіційної мови програмування для платформи Android на додаток до Java і C ++.

## 3.7 Середовище розробки VS Code

Visual Studio Code - це редактор вихідного коду, розроблений Microsoft для Windows, Linux та macOS. Він включає вбудований Git та підтримку налагодження, виділення синтаксису, інтелектуальне завершення коду, фрагменти та рефакторинг коду. Він налаштовується на високих розмірах, що дозволяє користувачам змінювати тему, комбінації клавіш, налаштування та встановлювати розширення, що додають додаткову функціональність. Вихідний код - вільний та відкритий, випущений згідно з дозвільною ліцензією MIT. Скомпільовані двійкові файли безкоштовно для будь-якого використання.

VS Code пори свою легкість має весь функціонал повноцінни IDE завдяки великій кількості плагінів.

## 3.8 Компоненти андроїд додатків

Компоненти програми є важливими складовими додатків для Android.

Ці компоненти слабко поєднуються з файлом маніфесту програми AndroidManifest.xml, який описує кожен компонент програми та спосіб їх взаємодії.

**Чотири основні компоненти**

Activities: діяльність являє собою один екран із користувальницьким інтерфейсом, стисла активність виконує дії на екрані. Наприклад, програма електронної пошти може мати одну активність, яка показує перелік нових електронних листів, інша діяльність для створення електронної пошти та інша діяльність для читання електронних листів. Якщо програма має більше ніж одну активність, то одну з них слід позначати як діяльність, яка представлена при запуску програми.

На (рисунку 3.4) зображений життєвий цикл activity.

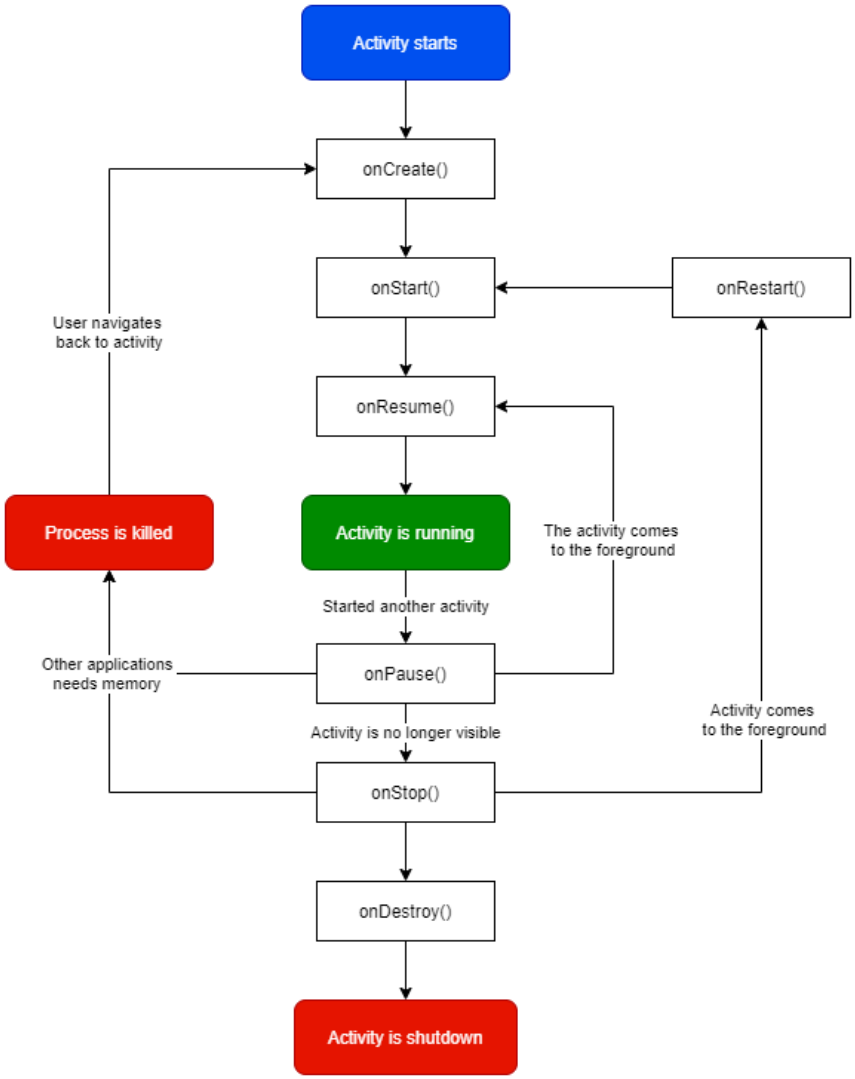


Рисунок 3.4 – життєвий цикл activity.

Services: служба - це компонент, який працює у фоновому режимі для виконання тривалих операцій. Наприклад, сервіс може відтворювати музику у фоновому режимі, поки користувач перебуває в іншому додатку, або він може отримувати дані по мережі, не блокуючи взаємодію користувача з діяльністю.

На (рисунку 3.5) зображений життєвий цикл сервісу.

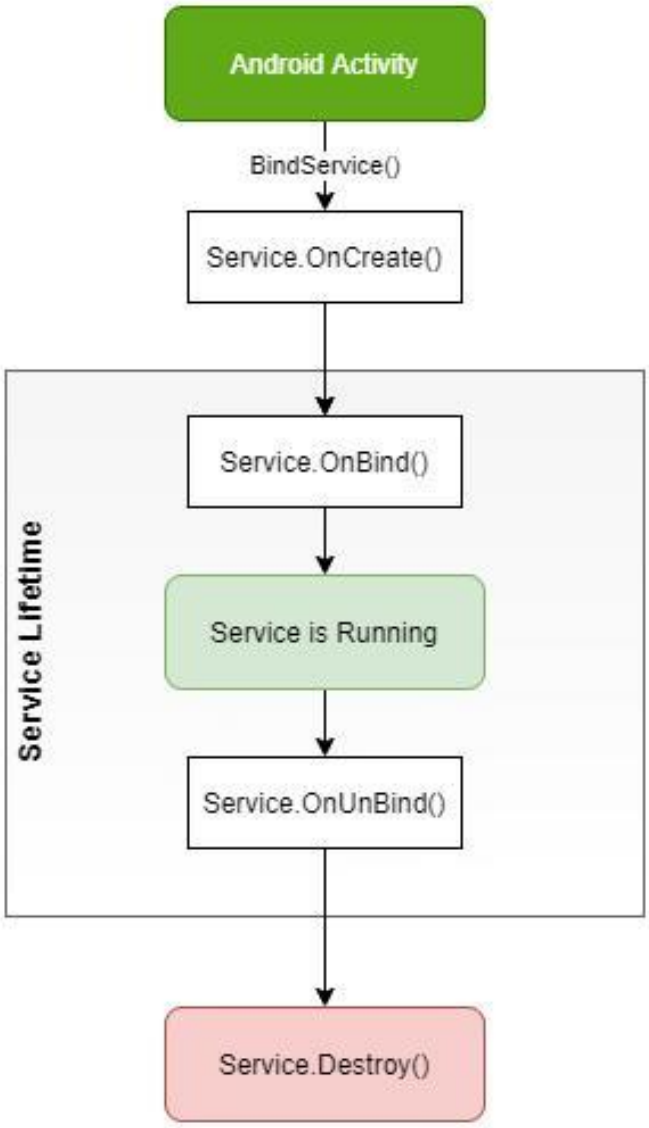


Рисунок 3.5 – життєвий цикл сервісу.

Broadcast Receivers: Транслятори мовлення просто відповідають на широкомовні повідомлення інших програм або з системи. Наприклад, програми також можуть ініціювати трансляцію, щоб іншим програмам було відомо про те, що деякі дані були завантажені на пристрій і вони доступні для використання, тому це транслятор, який перехопить це повідомлення та ініціює відповідні дії. Приймач широкомовної передачі реалізований як підклас класу BroadcastReceiver, і кожне повідомлення є транслятором як об'єкт Intent.

Content Providers: Компонент постачальника вмісту надає дані від однієї програми іншим на запит. Такі запити обробляються методами класу ContentResolver. Дані можуть зберігатися у файловій системі, в базі даних або в іншому місці. Постачальник контенту реалізується як підклас класу ContentProvider і повинен реалізувати стандартний набір API, що дозволяє іншим програмам здійснювати транзакції.

**Додаткові компоненти**

Fragments: представляє частину інтерфейсу користувача в діяльності.

Intents: повідомлення, що з'єднують компоненти разом.

Resources: зовнішні елементи, наприклад константи та зоюраження.

Views: елементи інтерфейсу, які малюються на екрані.

Layouts: перегляд ієрархій, які керують форматом екрана та виглядом.

Manifest: файл ресурсу, який містить усі деталі, необхідні андроїдній системі щодо програми. Це ключовий файл, який працює як міст між розробником android і платформою Android.

# 4. ОПИС РЕАЛІЗАЦІЇ

## 4.1 Архітектура програмної системи.

Реалізація андроїд додатку для розпізнавання режиму транспортування включає в себе модель, яка буде аналізувати дані отриманні з сенсорів мобільного телефону та повертати відповідний клас транспортування який буде відображати інтерфейс додатку. Для створення будь якої моделі машинного навчання потрібен набір даних. Було використано TMD dataset з Kaggle. Також було створено мобільний додаток для збирання даних з мобільних сенсорів що дозволяє доповнювати та тестувати модель даними зібраними власноруч.

На (рисунку 4.1) зображена загальна архітектура системи класифікації режиму пересування користувача.


Рисунок 4.1 – загальна архітектура системи.

На (рисунку 4.2) зображена архітектура програмного продукту.

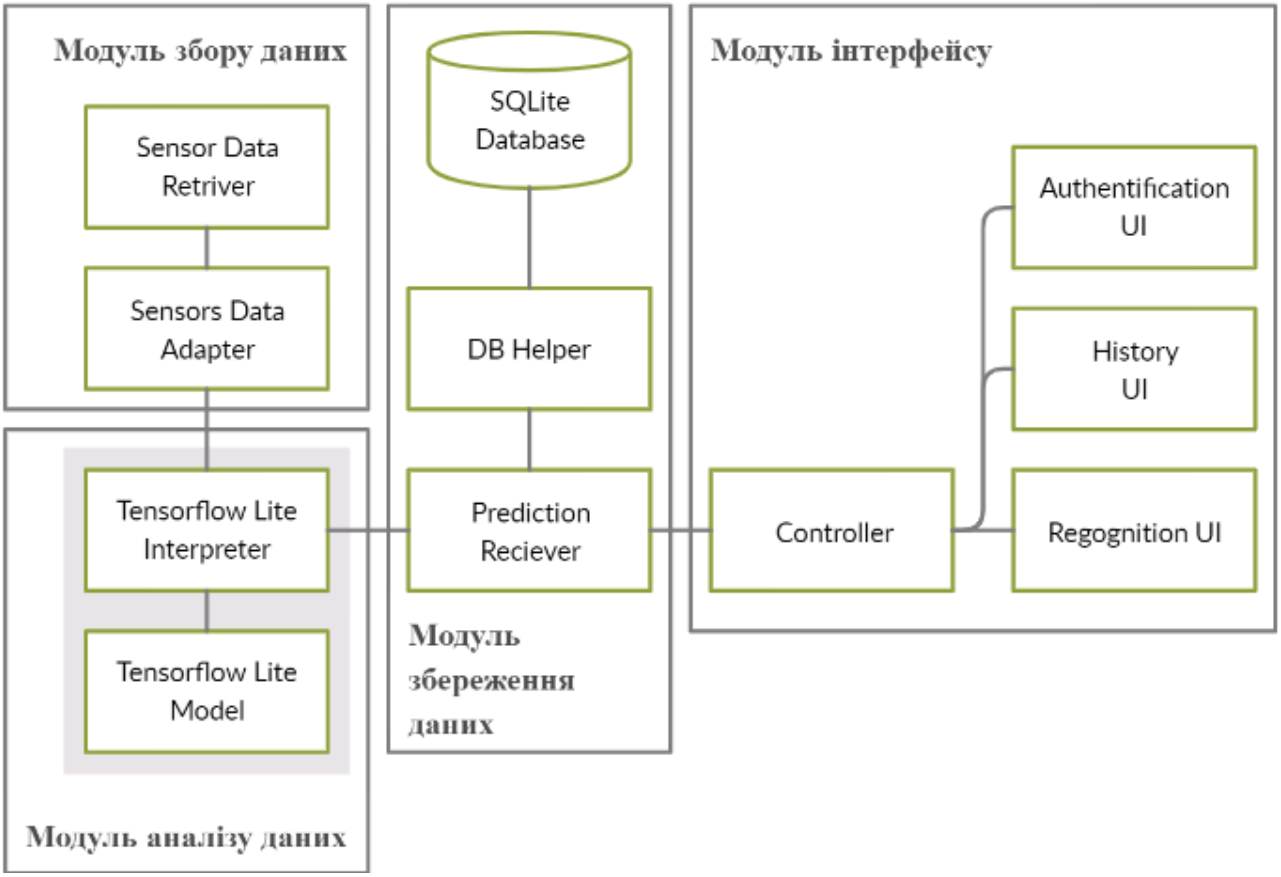


Рисунок 4.2 – архітектура програмного продукту.

Програмну систему можна умовно поділити на 3 модулі:

- модуль збору даних
- модуль аналізу
- модуль інтерфейсу
- модуль інтерфейсу

Модуль збору даних представлений блоками: Sensors Data Retriever, Sensor Data Adapter.

Модуль аналізу представлений блоками: TensorFlow Lite Interpreter, TensorFlow Lite Model. Варто виділити що блоки зображені на сірому фону тому що TensorFlow Lite Interpreter є фреймворком який використовує створену модель приймає данні та надає доступ до них.

На (рисунку 4.3) відображений цикл роботи мобільного додатку та створення моделі.

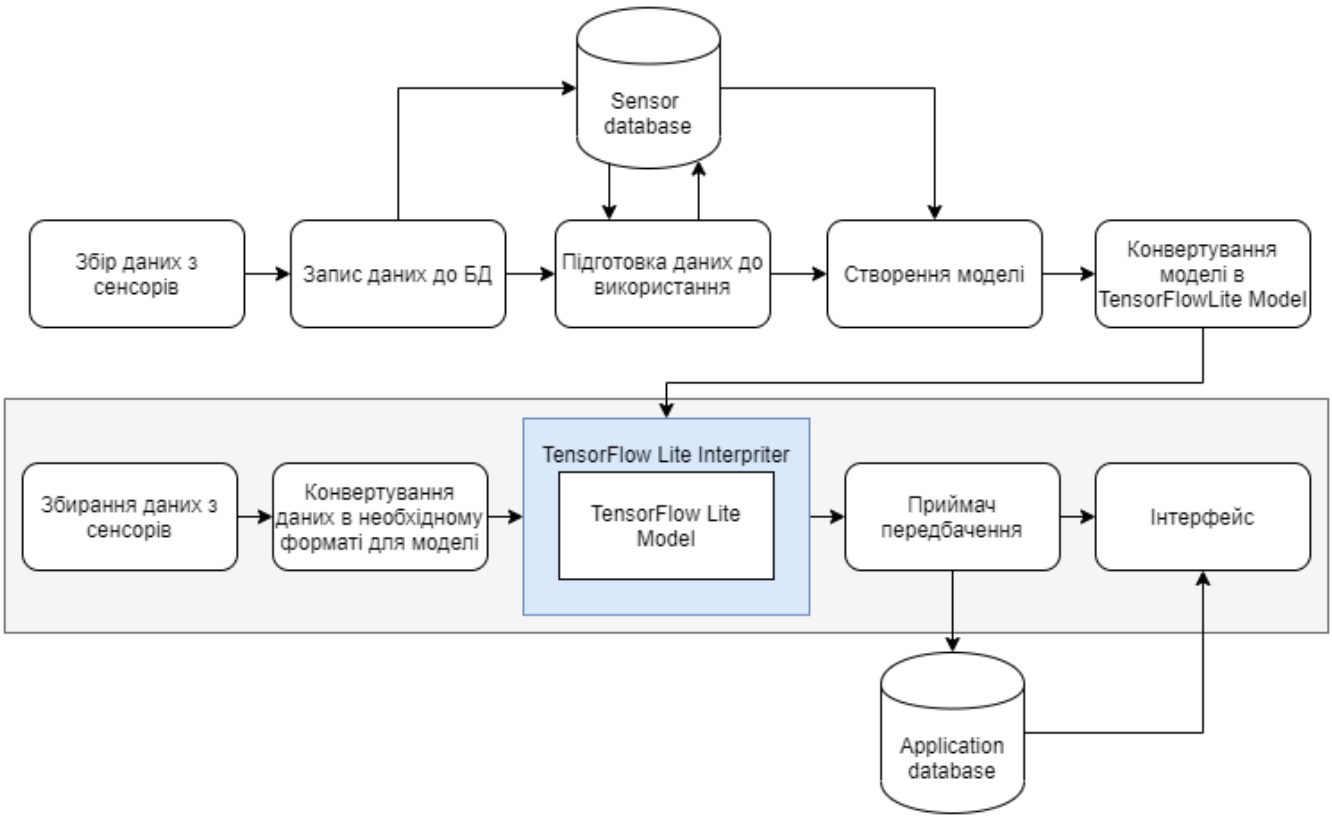


Рисунок 4.3 – цикл роботи мобільного додатку та створення моделі.

**Модуль збору даних**

Модуль збору даних збирає дані в відповідному форматі до побудованої моделі аналізу та не передає їх до наступного в потрібному до моделі форматі. Данні збираються з таких сенсорів:

- Accelerometer
- Sound
- Orientation
- Linear acceleration
- Speed
- Gyroscope
- Rotation vector

- Game rotation vector
- Gyroscope uncalibrated.

Формат даних який приймає модуль аналізу:

<timestamp, sensor, sensorOutput>

**Модуль аналізу**

Щоб використовувати модель в Android використали TensorFlow Lite converter з фреймворку TensorFlow Lite для того щоб конвертувати створену TensorFlow модель збережену в форматі SavedModel в TensorFlow Lite Model File(оптимізований формат файлу моделі на основі FlatBuffers).

Далі використали бібліотеку TensorFlow Lite interpreter з фреймворку TensorFlow Lite яка приймає файл моделі, виконує операції, визначені на вхідних даних, та забезпечує доступ до виводу.

Для забезпечення коректної роботи моделі модуль виконує перевірку формату вхідних даних.

Передбачення модель записує до бази даних з поміткою часу. Та передає інтерфейсу.

Схема дій модулю зображена на (рисунку 4.4).

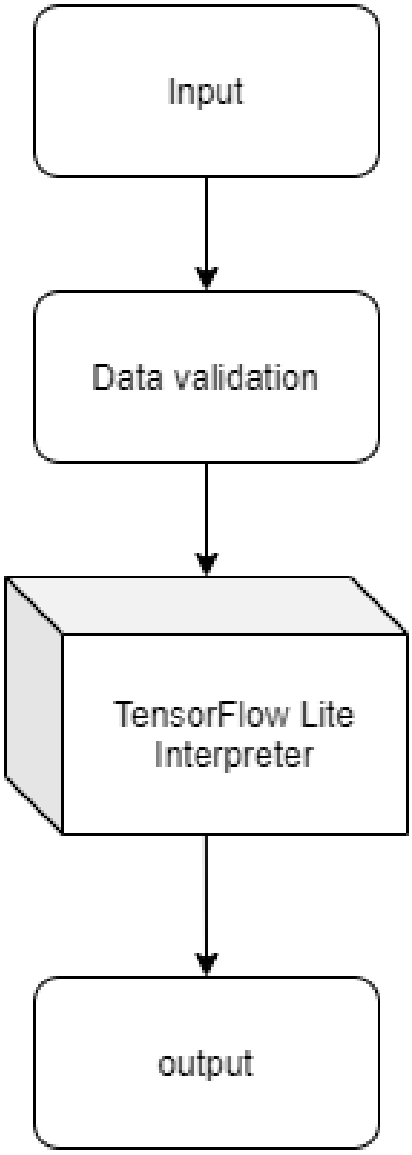


Рисунок 4.4 – схема обробки даних.

**Модуль інтерфейсу**

Створили інтерфейс для відображення результатів виконання моделі на основі зібраних даних. Інтерфейс отримує передбачення моделі щодо режиму транспортування та відображає його результат. Інтерфейс дозволяє користувачу авторизуватись та відстежувати режим транспортування.

Відстежувати режим транспортування можливо в реальному часі.

Також завдяки тому що активність користувача зберігається до бази даних з поміткою часу є можливим перегляд історії активності. В інтерфейсі це реалізовано в вигляди позначення витраченого часу на певну активність з початку відстеження до його закінчення

На (рисунку 4.5) зображений usecase інтерфейсу додатку.

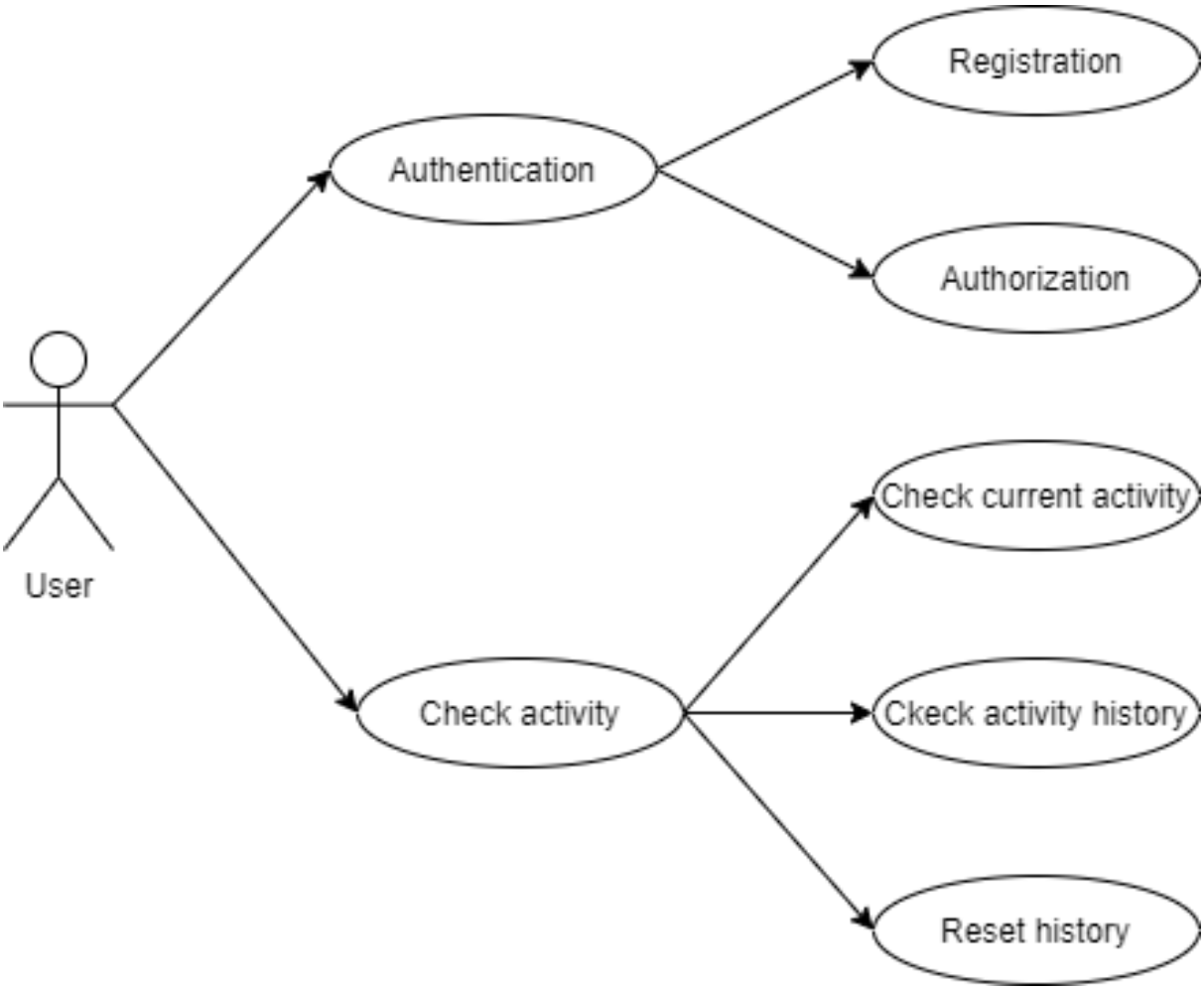


Рисунок 4.5 – usecase інтерфейсу.

Список Activity додатку:

- Login activity
- Registration activity
- Recognition activity
- History activity

На (рисунку 4.6) зображена карта переходів додатку між Activity.

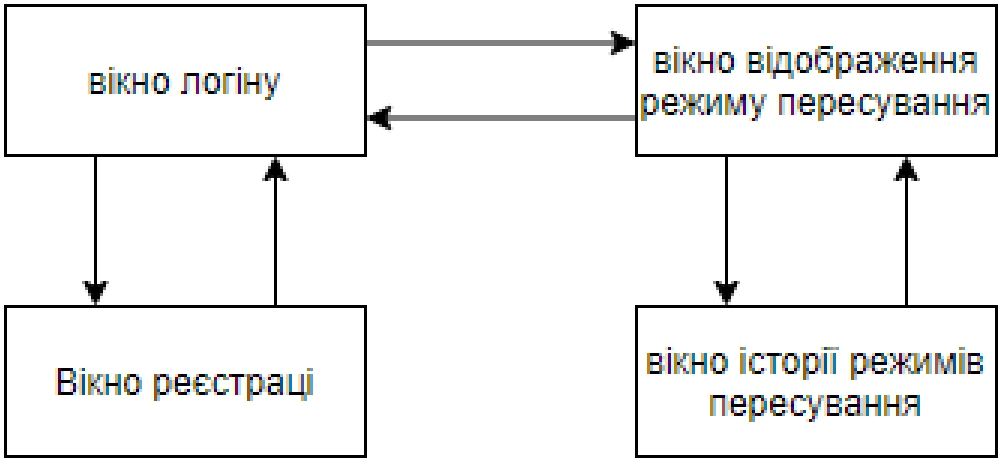


Рисунок 4.6 – карта переходів між Activity.

## 4.2 Опис бази даних

В програмному продукті створено 2 окремі бази даних. Перша база даних використовується андроїд додатком та зберігає інформацію про користувача та його активності, використовується для автентифікації та відображення історії активностей користувача . Друга база даних використовується середовищем створення моделі для пришвидшення процесу роботи з даними мобільних сенсорів. Містить данні сенсорів з позначками часу та назви активностей. Також використовується для створення наборів даних для навчання та тестування моделі.

Для створення та використання бази даних в андроїд додатку було використано полегшену реляційну систему керування базами даних SQLite.

Схема бази даних для андроїд додатку зображена на (рисунку 4.7).

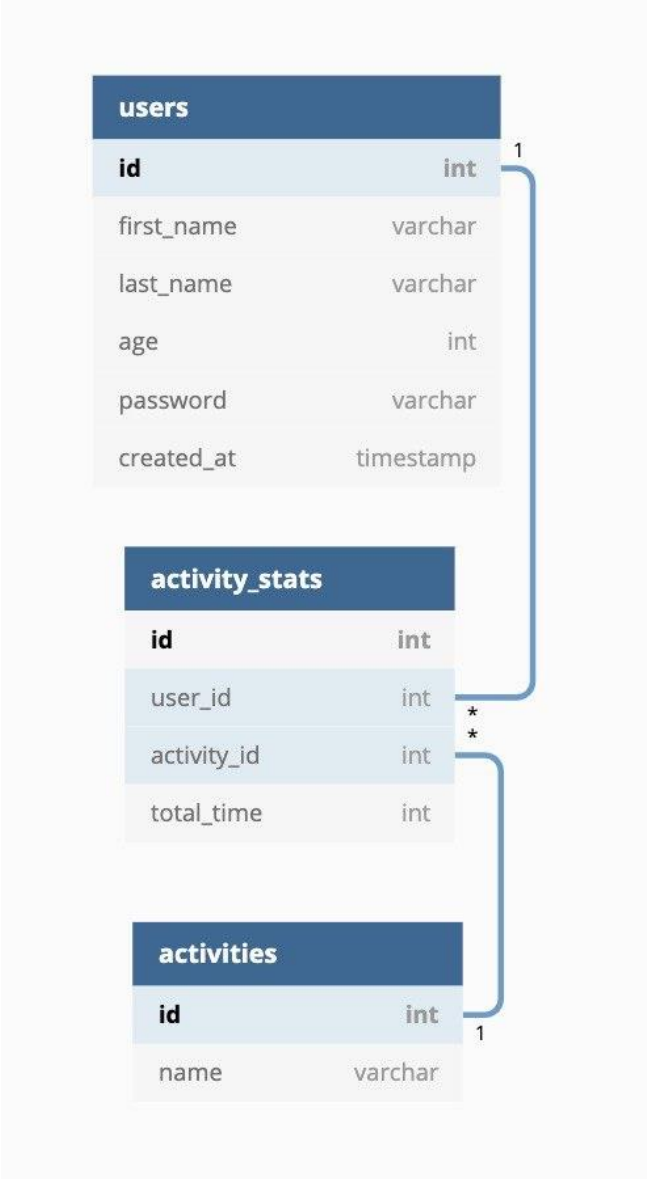


Рисунок 4.7 – схема БД для андроїд додатку.

Таблиця “USERS” (таблиця 4.1) містить в собі інформацію про користувача. Поля цієї таблиці використовуються для процесу автентифікації.

Таблиця 4.1. Структура таблиці “USERS”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Id | Int | Унікальне поле таблиці |
| First_name | String | Ім’я |
| Last_name | Stirng | Фамілія |

| | | |
|---|---|---|
| Age | String | Вік |
| Password | String | Пароль |
| Crearted_at | String | Дата реєстрації |

Таблиця “ACTIVITY_STATS” (таблиця 4.2) містить в собі інформацію активність та її час. Поля цієї таблиці використовуються для відображення історії активностей.

Таблиця 4.2. Структура таблиці “ACTIVITY_STATS”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| user_id | Int | Унікальне поле таблиці |
| activity_id | Int | Унікальне поле таблиці |
| total_time | Int | Зазальний час активності в секундах |

Таблиця “ACTIVITIES” (таблиця 4.3) містить в собі назви активностей.

Таблиця 4.3. Структура таблиці “ ACTIVITIES ”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Id | Int | Унікальне поле таблиці |
| name | String | Назва активності |

Схема бази даних для середовища створення моделі зображена на (рисунку 4.8).

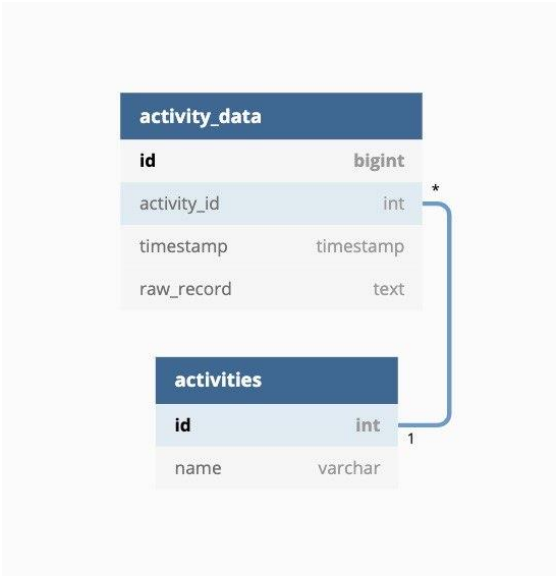


Рисунок 4.8 – схема БД для середовища створення моделі.

Таблиця “ACTIVITY_STATS” (таблиця 4.1) містить в собі інформацію зібрану з сенсорів.

Таблиця 4.1. Структура таблиці “ACTIVITY_DATA”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| activity_id | Int | Унікальне поле таблиці |
| Timestamp | String | Час запису даних |
| raw_record | String | Дані сенсорів |

Таблиця “ACTIVITIES” (таблиця 4.2) містить в собі назви активностей.

Таблиця 4.2. Структура таблиці “ ACTIVITIES ”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Id | int | Унікальне поле таблиці |
| Name | String | Назва активності |

## 4.3 Створення моделі машинного навчання

Структура датасету зображена на схемі(рисунок 9.9):

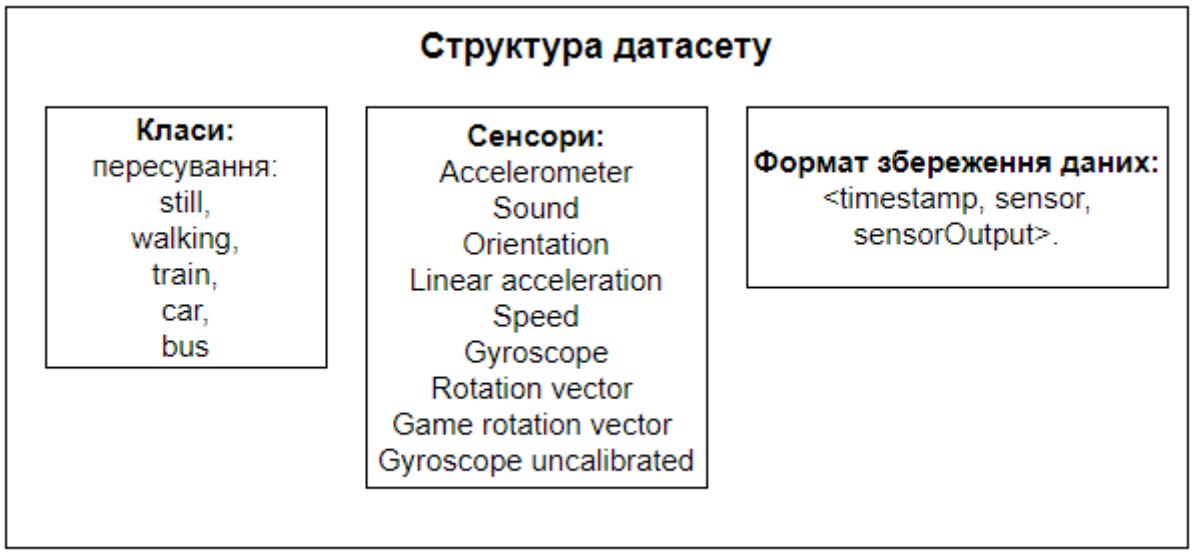


Рисунок 4.9 – структура датасету.

Загальний час запису даних(Total) 31:48:50, час запису даних з сенсорів для кожного класу окремо:

- Bus: 01:44:35
- Car: 07:53:50
- Still: 07:29:35
- Train: 06:20:25
- Walking: 08:20:25

Для доповнення дата сету було створено андроїд додаток який збирає данні з сенсорів та записує їх відповідному форматі до датасету.

На (рисунку 4.10) зображення інтерфейсу додатку.

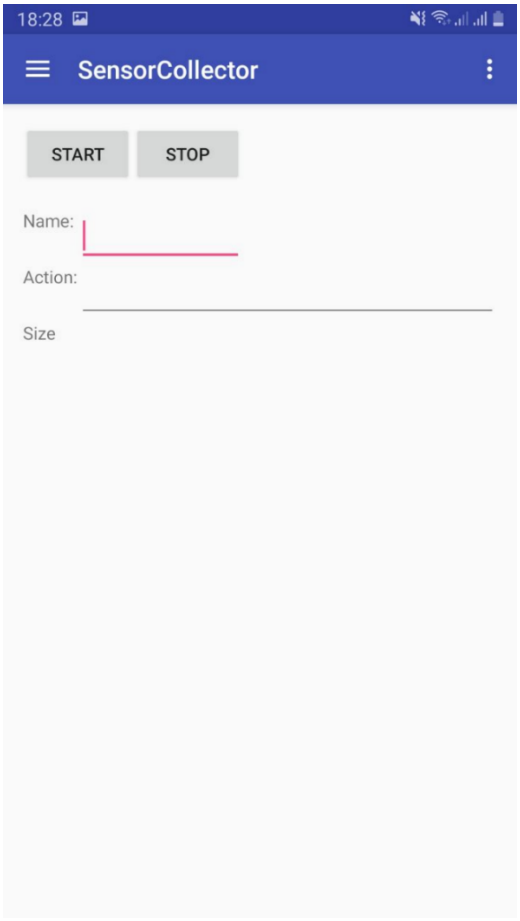


Рисунок 4.10 – інтерфейс додатку.

Процес очищення датасету:

- видалення пустих файлів
- видалення файлів з неправильними показниками датчиків

Деякі датчики повертають єдине значення, наприклад датчик звуку. Але частина датчиків повертають більш ніж одне значення які пов’язані з координатною системою. Тому впроваджена метрика (Магнітуда) для запису даних сенсорів одним значенням.

Далі дані розділяються на часові проміжки (вікна), в яких для кожного з проміжків генеруються:

- maximum,
- minimum,

- mean,
- standard deviation (число, яке використовується для визначення рівня розподілу вимірювань для групи із середнього чи очікуваного значення).

На (рисунку 4.11) зображений процес виділення статистичних даних з часових проміжків.

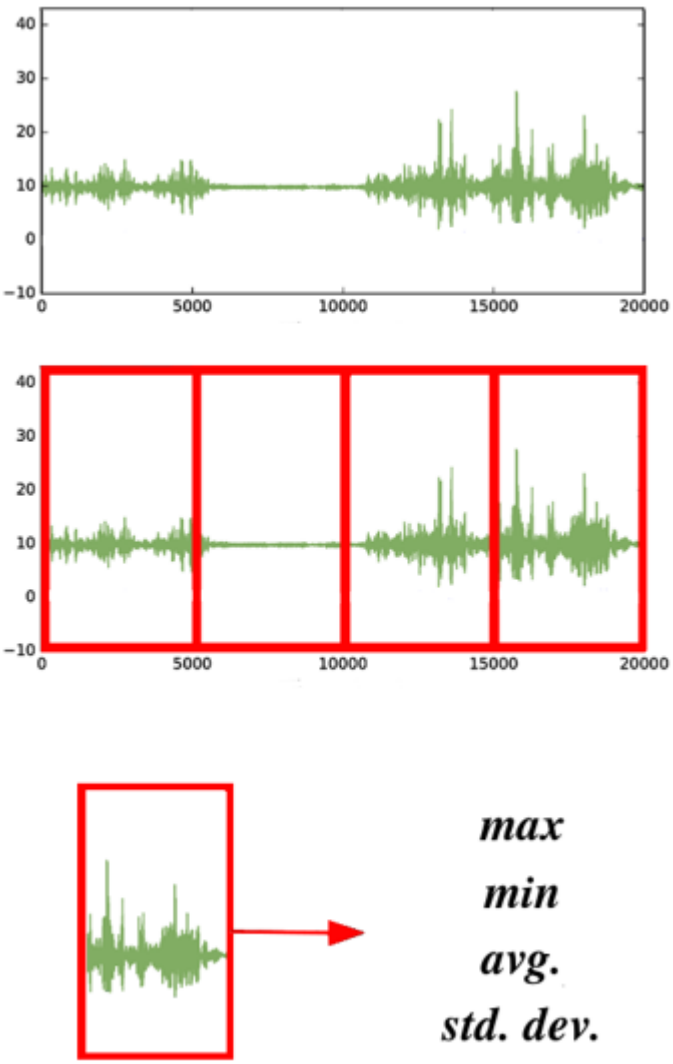


Рисунок 4.11 – Процес виділення статистичних даних з часових проміжків.

Спочатку створимо модель для кожного сенсору щоб проаналізувати ефективність та доцільність їх використання. Для побудови моделі використаємо

метод машинного навчання для класифікації Random forest. Результати зображенні на (рисунку 4.12).

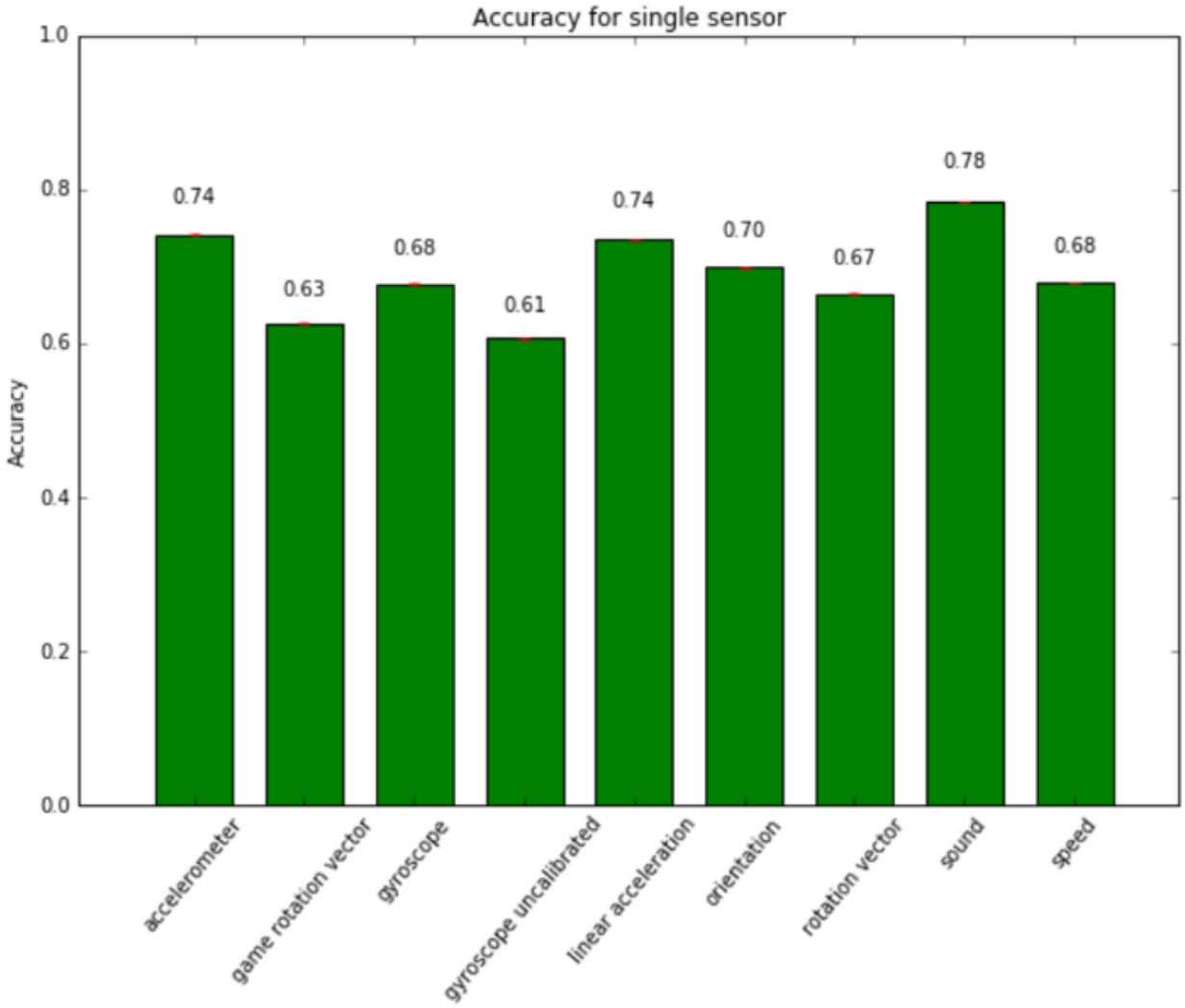


Рисунок 4.12 – результати тестування моделі.

Точність створеної моделі тільки з одним сенсором коливається від 0.57 до 0.78. Можна зробити висновок що всі сенсори наявні в датасеті можуть бути ефективно використанні для визначення режиму.

Цікаво що дані з звукового сенсору показали результат на одному рівні з сенсорами які найкраще зарекомендували себе в ідентифікації транспортних засобів акселерометру та гіроскопу.

Створимо 3 моделі машинного навчання для визначення найбільш ефективної.

Використані моделі: Random forest, Neural Network, Decision Tree.

Найбільш ефективним виявився метод Random forest з точністю 96%, Neural Network 95%, та Decision Tree 85%.

Створюємо SavedModel з методу Random forest для подальшого використання в додатку.

На (рисунку 4.13) зображено алгоритм процесу створення моделі.


Рисунок 4.13 – процес створення моделі.

# 5. РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

Для використання мобільного додатку потрібен андроїд смартфон з версією системи 4.0+.

Щоб встановити додаток потрібно запустити файл app.apk який розпочне встановлення додатку як зображено на (рисунку 5.1) та (рисунку 5.2).


Рисунок 5.1 – процес встановлення додатку.

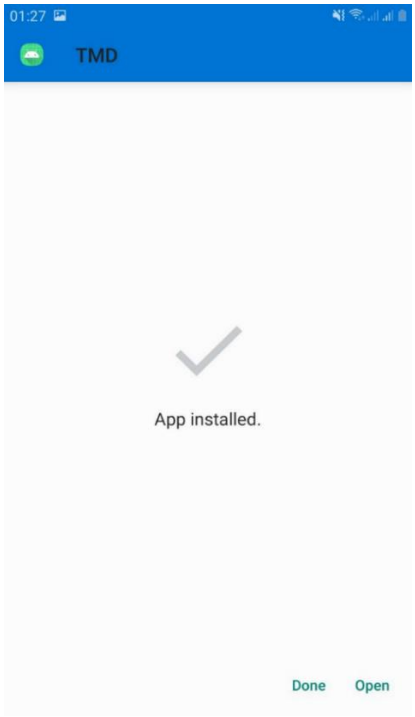


Рисунок 5.2 – процес встановлення додатку.

Після встановлення, якщо відкрити андроїд додаток з’явиться activity логіну(login activity) який при правильному введені логіну та паролю відкриває основну activity(recognition activity), при неправильному введені виводить повідомлення про помилку та залишається на поточній activity.

На (рисунку 5.3)  та (рисунку 5.4) зображена робота login activity.

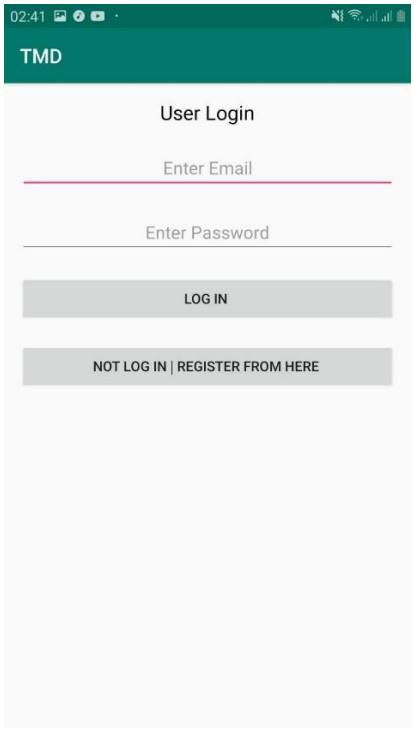


Рисунок 5.3 – приклад роботи login activity.

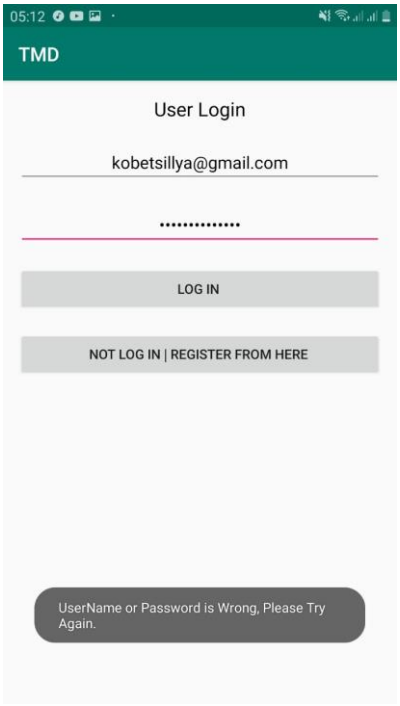


Рисунок 5.4 – приклад роботи login activity.

З login activity можливо перейти до registration activity якщо акаунт ще не зареєстрований або потрібен новий. При неправильному введені даних або повторному використанню почти буде виведено повідомлення про відповідну помилку.

На (рисунку 5.5)  та (рисунку 5.6) зображена робота registration activity.

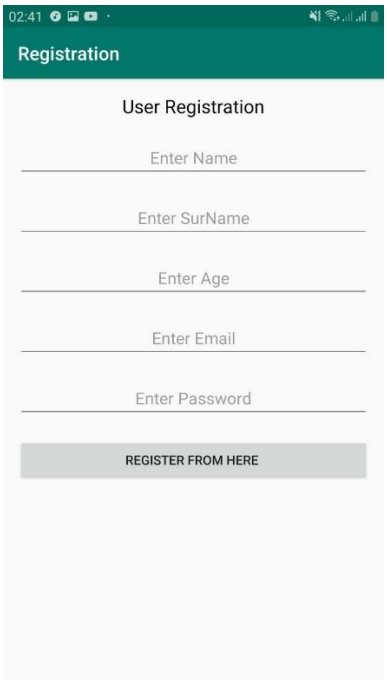


Рисунок 5.5 – приклад роботи registration activity.

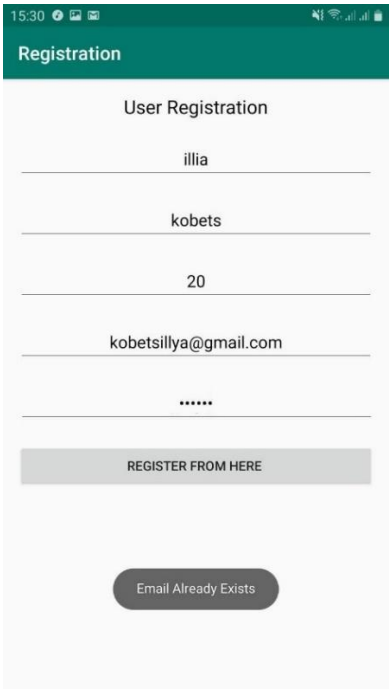


Рисунок 5.6 – приклад роботи registration activity.

Після успішного логіну програма відкриє recognition activity. Щоб розпочати процес або призупинити потрібно натиснути на кнопку “TRACK ACTIVITY”. Також натиснувши на ‘”ACTIVITY HISTORY” можна перейти до history activity.

На (рисунку 5.7) та (рисунку 5.8) зображені приклади роботи додатку.

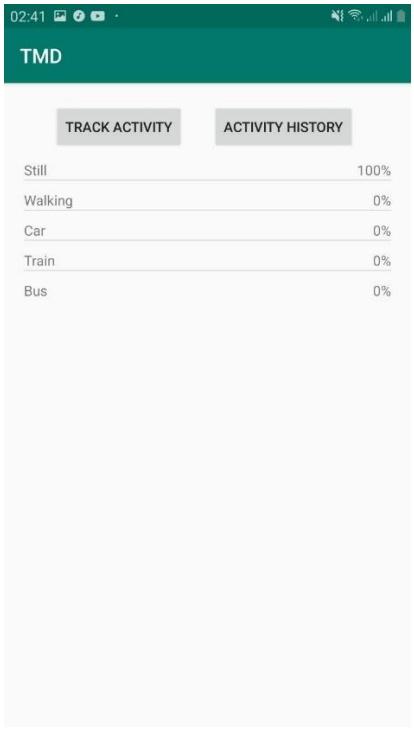


Рисунок 5.7 – приклад роботи recognition activity.

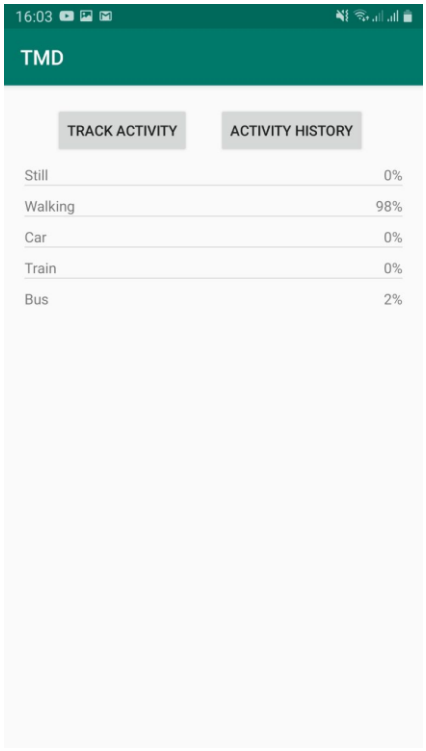


Рисунок 5.8 – приклад роботи recognition activity.

History activity відображає інформацію про кількість часу витрачену на кожен з класів акивностей. На (рисунку 5.9) зображена робота history activity.


Рисунок 5.9 – приклад роботи history activity.

# ВИСНОВКИ

Під час виконання дипломного проекту була досліджена тема інструментальних засобів ідентифікації транспортних засобів. Були розширенні знання в машинному навчанні а саме покращене знання бібліотек TensorFlow та Scikit-learn. Здобуті практичні навички в розробці андроїд додатків та побудуванні моделей машинного навчання. Покращене розуміння IoT та роботи мобільних сенсорів.

Буди зібрані дані мобільних сенсорів для використання створено модель машинного навчання для класифікації режиму пересування користувачі на основі цих даних.

Розроблена база даних для зберігання даних сенсорів телефону, та інших необхідних даних для роботи андроїд додатку.

Створено андроїд додаток для розпізнавання режиму пересування користувача в реальному часі та відображенні історію використаних режимів пересування на основі даних мобільних сенсорів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

**1.** Aurelien G. Hands-On Machine Learning with Scikit-Learn and TensorFlow / Geron Aurelien., 2017.

**2.** Hemminki S., Nurmi P., Tarkoma S.: "Accelerometer-based transportation mode detection on smartphones", SenSys 2013 Conf., Roma (Italy), 11-15 Nov. 2013

**3.** J. Yang. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In Proceedings of the 1st International Workshop on Interactive Multimedia for Consumer Electronics (IMCE), pages 1–9, 2009.

**4.** Y. Zheng, Y. Chen, Q. Li, and W.-Y. Xie, X. Ma. Understanding transportation modes based on gps data for web applications. ACM Transactions on the Web, 4,1, 2010.

**5.** Oriol Pujol and Petia Radeva. Human activity recognition from accelerometer data using a wearable device. Pattern Recognition and Image Analysis, 2011.

**6.** Ling Bao and Stephen S Intille. Activity Recognition from User-Annotated Acceleration Data. Pervasive Computing, 3001:1 17, 2004.

# ДОДАТОК 1

Інструментальні засоби ідентифікації транспортних засобів

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ62198_20Б

Аркушів 2

Київ – 2020

| Позначення | Найменування | Примітки |
|---|---|---|
| Документація | | |
| УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62202_20Б | Записка | Пояснювальна записка |
| Компоненти | | |
| УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62198_20Б 12-1 | Текст програмного модулю | Модуль створення моделі машинного навчання |
| УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ 62198_20 13-1 | Опис програми | Опис програми |

# ДОДАТОК 2

Інструментальні засоби ідентифікації транспортних засобів

Текст програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ62198_20Б 12

Аркушів 10

Київ – 2020

```
class TMDataset:
    def create_time_files(self):
        if self.sintetic:
            dir_src = const.DIR_SINTETIC_RAW_DATA_HEADER
            dir_dst = const.DIR_SINTETIC_RAW_DATA_FEATURES
        else:
            dir_src = const.DIR_RAW_DATA_HEADER
            dir_dst = const.DIR_RAW_DATA_FEATURES
        # create files with header if not exist
        if not os.path.exists(dir_src):
            self.create_header_files()
        if len(self.header) == 0 or len(self.header_with_features) ==
0:
            self.__fill_data_structure()
        if not os.path.exists(dir_dst):
            os.makedirs(dir_dst)
        else:
            shutil.rmtree(dir_dst)
            os.makedirs(dir_dst)
        print("DIVIDE FILES IN TIME WINDOWS AND COMPUTE FEATURES....")
        # build string header
        header_string = ""
        for i in self.header_with_features:
            header_string = header_string +
self.header_with_features[i] + ","
        header_string = header_string[:-1]
        header_string += ",target,user\n"
        # compute window dimension
        window_dim = int(const.SAMPLE_FOR_SECOND *
const.WINDOW_DIMENSION)
        # loop on header files
        filenames = listdir(dir_src)
        for current_file in filenames:
            if current_file.endswith("csv"):
```

```
                if not self.sintetic:
                    current_tm = current_file.split("_")[2]
                    current_user = current_file.split("_")[1]
                source_file_path = os.path.join(dir_src, current_file)
                df_file = pd.read_csv(source_file_path,
dtype=const.DATASET_DATA_TYPE)
                featureNames = [col for col in df_file.columns if
                                col != 'target' and col != 'user' and
col != 'time' and col != 'activityrecognition#0' and col !=
'activityrecognition#1']
                # max time in source file
                #print(current_file)
                end_time =
df_file.loc[df_file['time'].idxmax()]['time']
                #print(end_time)
                destination_file_path = os.path.join(dir_dst,
current_file)
                destination_file = open(destination_file_path, 'w')
                destination_file.write(header_string)
                start_current = 0
                i = 0
                # track previuos value, if no value are present for a
windows use previous
                previous_mean = []
                previous_min = []
                previous_max = []
                previous_std = []
                previous_activityRec = ""
                previous_activityRecProba = ""
                # loop on windows in file
                while True:
                    # current value for features
                    current_mean = []
                    current_min = []
                    current_max = []
```

```
                    current_std = []

                    # define time range
                    end_current = start_current + window_dim
                    if end_time <= end_current:
                        range_current = list(range(start_current,
end_time, 1))
                        start_current = end_time
                    else:
                        range_current = list(range(start_current,
end_current, 1))
                        start_current = end_current
                    # df of the current time window
                    df_current =
df_file.loc[df_file['time'].isin(range_current)]
                    nfeature = 0
                    if self.sintetic:
                        if df_current.loc[:, "target"].size > 0:
                            df_current_tm = df_current.loc[:,
"target"]
                            current_user = df_current.loc[:,
"user"].iloc[0]
                            equal = True
                            for tm in range(0,df_current_tm.size-1,1):
                                if not df_current_tm.iloc[tm] ==
df_current_tm.iloc[tm+1]:
                                    equal = False
                                    break
                            if equal == False:
                                continue
                            else:
                                current_tm = df_current_tm.iloc[0]
                    currentLine = ""
                    for feature in featureNames:
                        currentFeatureSerie = df_current[feature]
```

```
                        currentMean =
currentFeatureSerie.mean(skipna=True)
                        currentMin =
currentFeatureSerie.min(skipna=True)
                        currentMax =
currentFeatureSerie.max(skipna=True)
                        currentStd =
currentFeatureSerie.std(skipna=True)
                        if i == 0:
                            previous_mean.append(str(currentMean))
                            current_mean.append(str(currentMean))
                            previous_min.append(str(currentMin))
                            current_min.append(str(currentMin))
                            previous_max.append(str(currentMax))
                            current_max.append(str(currentMax))
                            previous_std.append(str(currentStd))
                            current_std.append(str(currentStd))
                        else:
                            if str(currentMean) == 'nan':

current_mean.append(str(previous_mean[nfeature]))
                            else:
                                current_mean.append(str(currentMean))
                            if str(currentMin) == 'nan':

current_min.append(str(previous_min[nfeature]))
                            else:
                                current_min.append(str(currentMin))
                            if str(currentMax) == 'nan':

current_max.append(str(previous_max[nfeature]))
                            else:
                                current_max.append(str(currentMax))
                            if str(currentStd) == 'nan':
```

```
current_std.append(str(previous_std[nfeature]))
                            else:
                                current_std.append(str(currentStd))
                            currentLine = currentLine +
str(current_mean[nfeature]) + ","
                            currentLine = currentLine +
str(current_min[nfeature]) + ","
                            currentLine = currentLine +
str(current_max[nfeature]) + ","
                            currentLine = currentLine +
str(current_std[nfeature]) + ","
                            nfeature += 1
                    if df_current.shape[0] > 0:
                        # select 'activityrecognition#0' and
'activityrecognition#1' from  df_current
                        df_current_google =
df_current[['activityrecognition#0', 'activityrecognition#1']]
                        df_current_google =
df_current_google[df_current_google['activityrecognition#1'] >= 0]
                        current_values = []
                        if df_current_google.shape[0] == 0:

current_values.append(previous_activityRec)

current_values.append(previous_activityRecProba)
                        else:
                            if df_current_google.shape[0] == 1:
                                df_row = df_current_google

current_values.append(df_row['activityrecognition#0'].item())

current_values.append(df_row['activityrecognition#1'].item())
                                previous_activityRec = ""
                                previous_activityRecProba = ""
```

```
                            else:
                                # pick prediction with max probability
to be correct
                                activity0 =
df_current_google.loc[df_current_google['activityrecognition#1'].idxma
x()][
                                    'activityrecognition#0']
                                activity1 =
df_current_google.loc[df_current_google['activityrecognition#1'].idxma
x()][
                                    'activityrecognition#1']
                                current_values.append(activity0)
                                current_values.append(activity1)
                                previous_activityRec = activity0
                                previous_activityRecProba = activity1

                    previous_mean = list(current_mean)
                    previous_min = list(current_min)
                    previous_max = list(current_max)
                    previous_std = list(current_std)
                    if len(currentLine) > 2:
                        line = str(i) + "," + str(current_values[0]) +
"," + str(current_values[1]) + "," + currentLine[

:-1]
                        line = line + "," + str(current_tm) + "," +
str(current_user) + "\n"
                        destination_file.write(line)
                    i += 1
                    if start_current == end_time:
                        break
        print("END DIVIDE FILES IN TIME WINDOWS AND COMPUTE
FEATURES......")

    def create_dataset(self):
```

```
        if self.sintetic:
            dir_src = const.DIR_SINTETIC_RAW_DATA_FEATURES
            dir_dst = const.DIR_SINTETIC_DATASET
            file_dst = const.SINTETIC_FILE_DATASET
        else:
            dir_src = const.DIR_RAW_DATA_FEATURES
            dir_dst = const.DIR_DATASET
            file_dst = const.FILE_DATASET
        # create files with time window if not exsist
        if not os.path.exists(dir_src):
            self.__create_time_files()

        if not os.path.exists(dir_dst):
            os.makedirs(dir_dst)
        else:
            shutil.rmtree(dir_dst)
            os.makedirs(dir_dst)
        filenames = listdir(dir_src)
        result_file_path = os.path.join(dir_dst, file_dst)
        with open(result_file_path, 'w') as result_file:
            j = 0
            for file in filenames:
                if file.endswith(".csv"):
                    current_file_path = os.path.join(dir_src, file)
                    with open(current_file_path) as current_file:  #
tm file
                        i = 0
                        for line in current_file:
                            # if the current line is not the first,
the header
                            if i != 0:
                                result_file.write(line)
                            else:
                                if j == 0:
                                    result_file.write(line)
```

```
                                i += 1
                        j += 1

    class TMDetection:
    dataset = Dataset()
    classes = []
    classes2string = {}
    classes2number = {}

    def get_sets_for_classification(self, df_train, df_test,
features):
        train, test = util.fill_nan_with_mean_training(df_train,
df_test)
        train_features = train[features].values
        train_classes = [self.classes2number[c] for c in
train['target'].values]
        test_features = test[features].values
        test_classes = [self.classes2number[c] for c in
test['target'].values]
        return train_features, train_classes, test_features,
test_classes

    def random_forest(self, sensors_set):
        features =
list(self.dataset.get_sensors_set_features(sensors_set))
        train_features, train_classes, test_features, test_classes =
self.__get_sets_for_classification(
            self.dataset.get_train, self.dataset.get_test, features)
        classifier_forest =
RandomForestClassifier(n_estimators=const.PAR_RF_ESTIMATOR)
        classifier_forest.fit(train_features, train_classes)
        test_prediction = classifier_forest.predict(test_features)
        acc = accuracy_score(test_classes, test_prediction)
        df_feature = pd.DataFrame(
```

```
            {'accuracy': acc, 'featureName': features, 'importance':
classifier_forest.feature_importances_})
        df_feature = df_feature.sort_values(by='importance',
ascending=False)

        print(str(features))
        print(str(classifier_forest))
        print("ACCURACY : " + str(acc))
        print("END RANDOM FOREST")
```

# ДОДАТОК 3

Інструментальні засоби ідентифікації транспортних засобів

## Опис програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ62202_20Б 13-1

Аркушів 8

Київ – 2020

# АНОТАЦІЯ

Метою цієї роботи є створення інструментального засобу ідентифікації транспортних засобів. Створено модель класифікації транспортних засобів на основі даних з сенсорів та андроїд додаток для визначення режиму транспортування.

# ЗМІСТ

# ЗАГАЛЬНІ ВІДОМОСТІ

Розроблена програмна система складається з моделі машинного навчання та андроїд додатку.

Система вирішує проблему ідентифікації транспортних засобів користувача. Програмний застосунок створений на базі оперативної системи андроїд тому для використання потребується андроїд пристрій.

Додаток для використання потрібно встановити на андроїд пристрій.

# ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблена програмна система вирішує проблему ідентифікації транспортних засобів використовуючи дані сенсорів та модель машинного навчання.

Мобільний пристрій працює як сканер сигналів сенсорів, які обробляє створена модель та надає доступ до передбачення режиму транспортування який потім відображається в інтерфейсі.

-6-

# ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Розроблена програмна система складається з мобільного додатку, алгоритм роботи користувача з додатком:

1. Автентифікації користувача в додатку
2. Перегляд режиму транспортування в реальному часі
3. Перегляд історії активностей користувача

# ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання програмної системи потрібно мати:

Мобільний пристрій, що базується на операційній системі Android, на який встановлюється додаток.

-8-

# ВХІДНІ І ВИХІДНІ ДАНІ

Вхідна інформація:

Дані з 9 сенсорів для 5 класів транспортування зібрані з андроїд пристроїв

Вихідна інформація:

Передбачення моделі щодо режиму транспортування.